Digital Communications and Networks 4 (2018) 277–286

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/en/journals/digital-communications-and-networks/

Semi-supervised multi-layered clustering model for intrusion detection

Omar Y. Al-Jarrah [a,*], Yousof Al-Hammdi [a], Paul D. Yoo [b], Sami Muhaidat [a], Mahmoud Al-Qutayri [a]

[a] Department of Electrical and Computer Engineering, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[b] Centre for Electronic Warfare, Information and Cyber (EWIC), Cranfield University, Defence Academy of the United Kingdom, Shrivenham, Swindon, SN6 8LA, United Kingdom

ARTICLE INFO

Keywords:
Semi-supervised intrusion detection
Machine learning
Classification
Ensembles
Big data

ABSTRACT

A Machine Learning (ML)-based Intrusion Detection and Prevention System (IDPS) requires a large amount of labeled up-to-date training data to effectively detect intrusions and generalize well to novel attacks. However, the labeling of data is costly and becomes infeasible when dealing with big data, such as those generated by Internet of Things applications. To this effect, building an ML model that learns from non-labeled or partially labeled data is of critical importance. This paper proposes a Semi-supervised Multi-Layered Clustering ((SMLC)) model for the detection and prevention of network intrusion. SMLC has the capability to learn from partially labeled data while achieving a detection performance comparable to that of supervised ML-based IDPS. The performance of SMLC is compared with that of a well-known semi-supervised model (tri-training) and of supervised ensemble ML models, namely RandomForest, Bagging, and AdaboostM1 on two benchmark network-intrusion datasets, NSL and Kyoto 2006+. Experimental results show that SMLC is superior to tri-training, providing a comparable detection accuracy with 20% less labeled instances of training data. Furthermore, our results demonstrate that our scheme has a detection accuracy comparable to that of the supervised ensemble models.

1. Introduction

As we head towards the Internet of Things (IoT) era, the number of devices that have the capability to collect and exchange data is increasing at a phenomenal rate. This is due to advances in semiconductors, networking, communications, sensors, and Internet-related technologies, which have resulted in ubiquitous connectivity to vast arrays of Internet-based infrastructures, services, and applications, such as banking and energy utility. Many applications in different fields, such as social networking, economy, healthcare, industry, and science, produce a huge amount of data, namely big data. In fact, it is predicted that as much data will be created as was created in the entire history of planet Earth, with 90% of current data being created in the last two years [1]. The emergence of big data combined with disappearing network boundaries and sophisticated attacks has elevated the risk of network intrusions. Maintaining the integrity and security of Internet-based services and infrastructures, particularly from cyber attacks, is paramount. For example, smart cities that are evolving based on IoT technologies will simply not function reliably without agile secure infrastructures.

An Intrusion Detection and Prevention System (IDPS) is an essential component of networks' security infrastructure, as it monitors, detects, and identifies potential intrusions. IDPSs are classified based on their ability to recognize attacks known and unknown [2]. Rule-based IDPSs make decisions based on rule sets defined by domain experts. Such IDPSs are successful in detecting known attacks, but they have limited capabilities in the case of novel attacks. Given the significant increase in network traffic, finding and coding rule sets of rule-based IDPSs are becoming difficult and time-consuming [3]. An anomaly-based IDPS builds a model of normality and considers a deviation from this model as an attack. Although anomaly-based IDPSs are shown to be capable of detecting novel/unknown attacks, pure training datasets of normal traffic are required to build their detection models. Collecting pure training datasets of benign/normal network traffic is difficult due to the high similarity between normal and malicious traffic.

Machine Learning (ML) algorithms have been adopted for IDPSs owing to their model-free properties, which allow them to learn complex malicious and normal models [2]. Although ML algorithms brought significant advantages to IDPSs by automating the generation process of the models/rules of detection, they have been deployed in limited scale in the real world [2,4] because supervised ML-based IDPSs require a

* Corresponding author.
E-mail addresses: omar.aljarrah@kustar.ac.ae (O.Y. Al-Jarrah), yousof.alhammadi@kustar.ac.ae (Y. Al-Hammdi), p.yoo@cranfield.ac.uk (P.D. Yoo), sami.muhaidat@kustar.ac.ae (S. Muhaidat), mqutayri@kustar.ac.ae (M. Al-Qutayri).

https://doi.org/10.1016/j.dcan.2017.09.009
Received 12 September 2017; Accepted 16 September 2017
Available online 22 September 2017

sufficient supply of labeled training data. Unfortunately, data labeling, which is normally performed by domain experts, is expensive in terms of time and cost [3]. In contrast, unsupervised ML-based IDPSs, such as clustering-based IDPS, build models with unlabeled data. However, the performance of unsupervised ML-based systems, in general, is not as good as the performance of supervised ML-based systems [5]. Ensemble ML models are known for having a better accuracy than individual classifiers. The basic concept of ensemble models is to reduce the overall model error by reducing the overall variance error based on the fact that classification error is composed of bias and variance errors. Basically, ensemble models improve the detection accuracy of the classification model by smoothing or averaging the overall variance of multiple classifiers with almost comparable bias [6]. Often, there is a trade-off relationship between the bias (*i.e.*, accuracy) and the variance (*i.e.*, precision) of a classifier where a classifier with low bias tends to have a high variance, and vice versa. Hence, ensemble models reduce the error due to variance, resulting in improved detection accuracy. However, the constituent classifiers of the ensemble model should have diversity in their variance error, otherwise, there would be no means of averaging. This requires each classifier to commit its error on a test instance independently from other classifiers. However, combining multiple classifiers does not necessarily produce a better result than that produced by the best classifier in the ensemble model; rather, it reduces the likelihood of choosing a poor-performing classifier [6]. Thus, with the ever-increasing size of data, there is a need for powerful unsupervised or semi-supervised learning algorithms that can perform the tasks of IDPS.

This paper introduces a Semi-supervised Multi-Layered Clustering (SMLC) model for the detection and prevention of network intrusion. The proposed model mitigates the deployment issues of existing supervised ML-based IDPSs as it can achieve comparable/better performance with partially labeled data. SMLC builds an ensemble model of multiple randomized layers using a K-Means algorithm. The local learning models of SMLC are learned from the resultant clusters at different layers. The final prediction of a test instance is obtained by choosing the classification with the most votes among all decisions from all layers. The contributions of this work are as follows:

- Design and development of a semi-supervised model for network intrusion detection tasks. The proposed model utilizes the fact that instances of the same class type stay close in the Euclidean space to reduce data labeling errors, which improves the final detection accuracy (Section 3).
- Relying on the concept of the weighted Euclidean distance measure and atomic clusters, we argue that the time to update the model and classification time of the proposed model can be significantly reduced by building binary classifiers at non-atomic clusters only (Section 3.1).
- Comparisons of SMLC with well-known supervised ensemble ML models, namely RandomForest (RF), Bagging, and AdaBoostM1, and a semi-supervised model, the tri-training algorithm, in terms of model accuracy, detection rate, false alarm, Matthews correlation coefficient, training time, and testing time on two benchmark network intrusion datasets, NSL and Kyoto 2006+ (Section 4).

The remainder of this paper is organized as follows. Section 2 provides an overview of the current development of ML models for IDPS. Section 3 describes the proposed SMLC model. Section 4 discusses the settings of the experiments and the results. Section 5 presents the conclusions of this work.

2. Related work

ML refers to computer algorithms that learn from experiences without being programmed. An ML algorithm takes data of the instance space as an input and outputs a hypothesis of a defined hypothesis space that describes regularities in the data [2]. Supervised ML algorithms build learning models on training datasets of paired input instances and their corresponding labels or outputs. On the other hand, unsupervised algorithms group input instances into clusters based on some similarity measures. It is worth noting that the ability of an ML algorithm to learn the underlying patterns in training data and generalize to unseen events depends on the quality and quantity of the training data [7]. Recently, combinations of ML techniques, which are also known as ensemble models, have gained significant attention in the ML community as they often perform better than individual models and adapt quickly to new concepts [8–10]. Basically, an ensemble model generates multiple base classifiers that commit an error on identical data patterns independently. The final verdict of the ensemble model is derived from the individual predictions of the constituent base classifiers. Clustered ensemble [11], Bagging [12] and Boosting [13] are well-known ensemble models. Unlike previously mentioned ML models, semi-supervised models use both labeled and unlabeled data to build their final hypothesis [14,15]. Several studies in the literature, such as [16–20], have adopted semi-supervised learning approaches for the detection and prevention of intrusion.

Generally speaking, semi-supervised classification techniques can be categorized into self-training, co-training and multi-view learning, and generative models and graph-based methods [21]. Self-training methods use their own predictions to update their learning models. Given data of labeled and unlabeled data points, initially, a self-training method builds a learning model using the labeled portion of the data. The learned model is used to predict the label of unlabeled data points in the data. The data points with best predictions (*i.e.*, high confidence) are augmented with the labeled portion of the data and used to rebuild the learning model. This process is performed repeatedly until all unlabeled data points are used. Co-training methods assume that the input features can be split into two sufficient, conditional independent views [22]. Such methods build two classifiers (h_1,h_2) using the independent views of the training data. The most confident predictions of each classifier on the unlabeled data are then used to iteratively generate additional labeled data points for the other classifier. Each classifier is retrained iteratively using the labeled data points and the data points obtained from the other classifier. Generative models assume that a model $p(x,y)$ is an identifiable mixture distribution where $p(x,y) = p(x)p(x|y)$ [23]. In large scale data, only one labeled data point of every component is required to identify the mixture distribution. Graph-based methods build graphs where nodes represent data points (labeled or unlabeled) and edges represent the similarity among the data points [23].

Wagh and Kolhe [16] presented a semi-supervised approach to intrusion detection. The approach uses the most confident filtered data from a test dataset to refine the existing training dataset, which is used automatically to train the system again. Chen et al. [17] proposed two semi-supervised classification methods, Spectral Graph Transducer and Gaussian Fields Approach, to detect unknown attacks and one semi-supervised clustering method, MPCK-means. Li et al. [18] proposed an intrusion-detection algorithm based on semi-supervised fuzzy clustering in which a few labeled instances are used as seeds to initialize the classifier of the system. Chiu et al. [19] introduced a semi-supervised learning mechanism to build an alert filter that reduces false alarms and keeps a high detection rate. The mechanism uses Two-Teachers-One-Student (2T1S) as a learner for the proposed ML engine. Meng et al. [20] applied a disagreement-based semi-supervised learning algorithm to construct a false alarm filter and investigated its performance on alarm reduction in a network environment. Ashfaq et al. [24] proposed a novel fuzziness-based semi-supervised learning approach where a single hidden layer feed-forward neural network is trained to output a fuzzy membership vector. A sample categorization on unlabeled datapoints is then performed using the fuzzy quantity. The classifier is retrained after including each category separately in the original training data. Zhu and Fang [25] proposed a semi-supervised intrusion detection algorithm based on the concept of Natural Neighbor (2 N). The proposed algorithm performs clustering on labeled

data and then classification on unlabeled data according to the result of clustering. Xiang et al. [26] proposed an incremental semi-supervised training framework that combines the low space complexity advantage of topology learning and semi-supervised learning for network intrusion detection. Duong and Hai [27] proposed a semi-supervised model using a modified Mahanalobis distance based on Principle Component Analysis (PCA) for network traffic anomaly detection. Mousavi et al. [28] proposed a novel online version of the Laplacian twin support vector machine classifier, which can exploit the geometry information of the marginal distribution embedded in unlabeled data.

Contrary to the co-training-based IDPS introduced in Ref. [29], the proposed SMLC model does not require the generation of different views of the data. More importantly, SMLC presents a new methodology to generate a semi-supervised ensemble model. In addition, SMLC does not place any constraints on the used supervised algorithm. The SMLC model augments the learning process of its local learning models by 1) utilizing the unlabeled data and 2) dividing the training data into mutually exclusive clusters at each layer, exploiting the fact that instances of the same class type tend to stay close to each other in Euclidean space. SMLC enhances the overall detection accuracy by 1) providing diversity among its base classifiers as it generates multiple randomized layers using K-Means clustering, 2) identifying difficult- and easy-to-classify instances, and 3) employing most votes to arrive at the final prediction of an instance.

3. Proposed semi-supervised multi-layered clustering model

In this section, we present our SMLC model based on the concept of data clustering. SMLC follows the work presented in Ref. [30] and exploits the fact that resultant clusters of the K-Means algorithm depend on the initialization parameters (*e.g.*, seed, number of clusters) to provide diversity among its base classifiers [30]. Data instances might be assigned to different clusters when different initialization parameters are used [31]. In this context, a layer is defined as an object of K-Means using a randomized parameter (*i.e.*, seed). Thus, a data point/instance might belong to different clusters at different layers. Clusters at different layers might have overlapping but non-identical data, but clusters at the same layer are mutually exclusive and might have data points of one or multiple classes. To clarify this, consider a dataset of two classes, as shown in Fig. 1. (a). In this context, as shown in Fig. 1. (b), a layer is defined as an object of a K-Means clustering algorithm based on one set of initialization parameters (*i.e.*, seed). Thus, a data point/instance might belong to different clusters on different layers, as shown in Fig. 1. (b) and (c). Clusters at the same layer are mutually exclusive (*i.e.*, non-overlapping). However, clusters at different layers might have overlapping but non-identical data, as shown in Fig. 1. (d). Hence, clusters at different layers might be used to build diverse base classifiers, which can be used to construct an ensemble model that covers the whole decision space, as shown in Fig. 1(d).

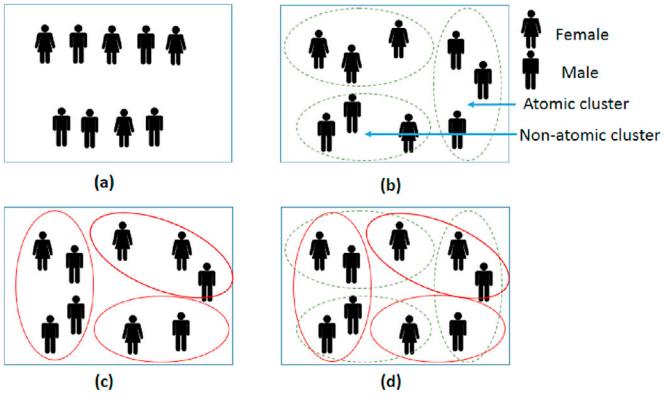

Fig. 1. (a) a dataset of two classes (male and female); (b) resultant clusters of K-Means algorithm using seed 1 (i.e, layer 1); (c) resultant clusters of K-Means algorithm using seed 2 (*i.e.*, layer 2); (d) a projection of the resultant clusters at different layers.

SMLC generates multiple layers of the randomized K-Means algorithm. A training dataset is fed to all layers in the SMLC model and portioned into K clusters at each layer. The resultant clusters at each layer might have data points/instances of different classes or one class. We refer to the cluster of instances of one class type as an atomic cluster (*e.g.*, cluster 3 in Fig. 1. (b)), whereas a non-atomic cluster is defined as the cluster of instances of multiple classes (*e.g.*, cluster 2 in Fig. 1. (b)). In order to infer dependencies of the data, SMLC builds local learning models on the resultant clusters at each layer. Basically, it builds binary classifiers at non-atomic clusters and remembers the class label of atomic clusters. In the following subsections, we describe 1) the data clustering using K-Means and a weighted Euclidean distance, 2) the training, 3) and the testing processes of the SMLC model.

3.1. Data clustering and weighted euclidean distance

The K-Means clustering algorithm is well known and widely used owing to its simplicity and ease of implementation [32]. However, the resultant clusters of K-Means depend heavily on the initialization parameters. SMLC exploits this property to provide diversity among its base classifiers. It also builds a local learning model on the generated overlapping but non-identical clusters at different layers.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be a set of instances in d-dimensional space and K is a predefined number of clusters. The K-Means algorithm minimizes the objective function given by Ref. [31]:

$$F(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in c_k} \| \mathbf{x}_i - \overline{\mathbf{x}_k} \|^2 \tag{1}$$

where c_k denotes the *kth* cluster,

$$\overline{\mathbf{x}_k} = \frac{1}{n_k} \sum_{\mathbf{x}_i \in c_k} \mathbf{x}_i \tag{2}$$

is the center of the *kth* cluster, and n_k is the number of instances in the *kth* cluster. $\| \cdot \|$ denotes the Euclidean norm used by the K-Means algorithm. The algorithm starts with K instances that represent the centroids of the clusters. Each instance in the training dataset is assigned to the centroid of the closest cluster and the mean of the instances in the same cluster is calculated. The procedure is repeated iteratively until convergence or an exit condition is satisfied.

Distance measure is an important aspect to consider in the application of the K-Means algorithm. Euclidean distance is the most widely used distance measure, and is given by the following equation [31]:

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{j=1}^{d} (x_j - z_j)^2} \tag{3}$$

where d is the number of dimensions or features. x_j and z_j are the values of the *jth* attribute of $\mathbf{x}$ and $\mathbf{z}$, respectively. Although the Euclidean distance works well for clusters with spherical homogeneous covariance matrices, it still treats all attributes equally when computing the distance between instances. Such an approach is not desirable when some attributes are more important to discriminate between patterns [33]. As such, deploying such measures might yield a low performance and affect the required number of iterations until the convergence of the K-Means algorithm. Therefore, we introduce the use of a weighted Euclidean distance measure based on the observation that different attributes might have a strong impact on the resultant partitions of data. The weighted Euclidean distance assigns a weight for each attribute based on its significance in distinguishing between class types. These weighted attributes can lead to a higher probability of obtaining atomic clusters with a

lower value of K (*i.e.*, number of clusters), which have the following advantages:

1. Reducing the overall complexity of the proposed model. This is mainly due to the fact that SMLC uses the class labels of atomic clusters only, eliminating the need for building binary classifiers.
2. Increasing the prediction efficiency of the system, since the overall number of binary classifiers used is reduced. In this case, the test instances are examined by fewer binary classifiers, which reduce the testing time per instance.

In this paper, the Information Gain Ratio (*IGR*) [34] is used as an attribute's weight, since it reflects the utility and significance of the attribute in detecting a class type, and it is given by

$$IGR(Y,A_j) = \frac{H(Y) - H(Y|A_j)}{H(A_j)} \tag{4}$$

where Y is the class and A_j is the *jth* attribute. Here, $H(.)$ is the entropy function given by

$$H(X) = -\sum_{\forall i} P(x_i)\log_2[P(x_i)] \tag{5}$$

where $P(.)$ is the probability operator and i is an index of the probabilities in a given input. Hereafter, the proposed weighted Euclidean distance based on *IGR* is given by

$$d(\mathbf{x},\mathbf{z}) = \sqrt{\sum_{j=1}^{d} w_j(x_j - z_j)^2} \tag{6}$$

where w_j denotes the weight of the *jth* attribute. The weights of the attributes are calculated and then passed to each layer in SMLC. The value of *IGR* of the *jth* attribute is assigned to w_j as follows:

$$w_j = IGR(Y,A_j) \tag{7}$$

3.2. Training process of SMLC

SMLC deals with partially labeled data. In this case, the training data include labeled and unlabeled instances. Each labeled instance is given a class label. On the other hand, unlabeled instances are not given any label. Let the training dataset be denoted by $T = \{T_{Labeled}, T_{Unlabeled}\}$, where $T_{Labeled} = \{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n)\}$, n denotes the number of labeled instances in $T_{Labeled}$, $T_{Unlabeled} = \{(\mathbf{x}_{n+1}),(\mathbf{x}_{n+2}),\ldots,(\mathbf{x}_N)\}$, N is the number of instances in T, $\mathbf{x}_i$ denotes the *ith* instance (*i.e.*, $\mathbf{x}_i =< x_{i1}, x_{i2}, \ldots, x_{id} >$), d represents the number of attributes, and $y_i \in Y$ is the class label set. Fig. 2 shows the stepwise procedure of SMLC. The training process of SMLC has two main phases:

1. SMLC generates overlapping but diverse clusters at different layers by using different initialization parameter sets (*i.e.*, random seed). The K-Means generates diverse clusters at different layers, $\{C_{1,2},\ldots,C_{1,K}\},\ldots,\{C_{L,2},\ldots,C_{L,K}\}$. Here we modified the K-Means algorithm to use the distance measure expressed by (6). w_j is calculated according to (7) and its value is obtained from the *IGR* analyzer. It should be noted that the *IGR* analyzer uses $T_{Labeled}$ only to calculate the weights of the attributes. The class labels of the training instances have not been considered in data clustering because the training dataset contains labeled and unlabeled instances.
2. The resultant cluster could be one of three types: a) fully labeled cluster that contains labeled instances only; b) partially labeled cluster that contains labeled and unlabeled instances; and c) unlabeled cluster. During the second phase, SMLC identifies fully labeled, partially labeled, and unlabeled clusters and builds a learning model on each cluster as follows:

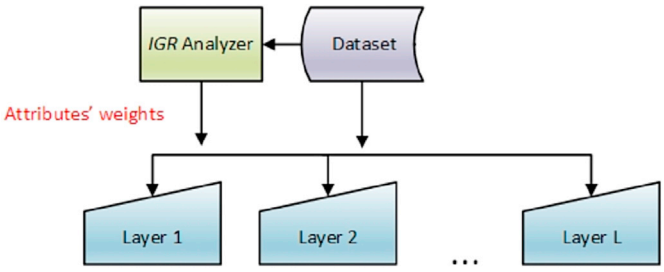

Fig. 2. Stepwise procedure of SMLC.

- In the case of fully labeled clusters, SMLC identifies atomic and non-atomic clusters at each layer. Equation (8) illustrates how a class distribution function is calculated for each cluster $C_{l,k}$:

$$F_{C_{l,k}}(y_j) = \sum_{\forall(\mathbf{x}_i,y_i)\in C_{l,k}} \theta(y_j,y_i) \tag{8}$$

where

$$\theta(y_j,y_i) = \begin{cases} 1 & \text{if } y_j = y_i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The cluster $C_{l,k}$ is defined as atomic if it satisfies the following equation:

$$\frac{Max_{\forall y_j\in C_{l,k}}\left(F_{C_{l,k}}(y_j)\right)}{\sum_{\forall y_j\in C_{l,k}} F_{C_{l,k}}(y_j)} = 1 \tag{10}$$

Thereafter, the SMLC builds binary classifiers, and C4.5 decision trees [35] in this study, with non-atomic clusters, and remembers the class label of atomic clusters. A decision tree is a graphical representation of possible decisions through a sequence of certain conditions or tests. It consists of nodes, edges, and branches. Nodes are arranged in a tree hierarchy and represent a test on an attribute, based on which the data is partitioned; each branch represents the outcome of the test, and each leaf node represents a class label or decision. Each node has a number of edges, which are labeled according to the possible value of the test and connect between nodes. A root node represents the topmost node and has no incoming edges. Decision trees take labeled training data, which might contain numerical or categorical values, as an input, and construct decision models that can be used to predict the class type of a new instance or data point [55].

- SMLC builds tri-training models with partially labeled clusters, which contain labeled and unlabeled instances, at each layer. A tri-training model builds three non-identical classifiers with the labeled instances in each partially labeled cluster [36]. The three classifiers are then refined using the unlabeled instances in the partially labeled cluster. In each iteration of the tri-training, an unlabeled instance is labeled for a classifier if the other two classifiers agree on the labeling [36]. Unless the three base classifiers are drawn from different distributions, they will always agree on the class label of the unlabeled instance. Therefore, the base classifiers are initially trained on bootstrapped training datasets from the labeled instances. The final hypothesis is produced via majority voting among all individual decisions of the three base classifiers. Please refer to [36] for more details on how tri-training works.
- The resultant clusters might contain unlabeled instances only. In that case, at each layer, SMLC finds the nearest neighbor labeled instances from the labeled portion of the training dataset ($T_{Labeled}$) to the centroids of the unlabeled clusters. Then, it combines the unlabeled clusters with its corresponding labeled data to form a new dataset of labeled and unlabeled instances. Finally, SMLC builds tri-training models on the newly formed partially labeled clusters, as described above. Fig. 3 shows a flowchart for building local learning models at each layer of the SMLC model.

3.3. Testing process of SMLC

The testing process of a test instance begins with finding the nearest clusters' centroid at each layer of SMLC. The cluster that has the minimum distance between its centroid and the testing instance at each layer is selected as the appropriate cluster. Then, the correspondent classifier at that layer is used to predict the class type of the testing instance. The final label of the testing instance is determined by the corresponding classifiers with the most votes at different layers.

4. Evaluation and analysis

Thorough evaluation of an IDPS is of crucial interest as many approaches fail to meet expectations in real-world scenarios [4]. This requires an appropriate dataset that represents the real-world scenario. The most widely used datasets are DARPA/Lincoln packet traces [37, 38] and KDD Cup [39] which is derived from them. Tavallaee et al. [40] statistically showed that the original KDD Cup dataset has some shortcomings. For example, the KDD Cup dataset is heavily imbalanced to attack examples. Approximately 80% of examples are attacks; but a typical network contains approximately 0.01% attacks. Thus, the KDD Cup representation of a real-world scenario has been criticized. In addition, the KDD Cup dataset contains many duplicate and redundant records. Tavallaee et al. [40] introduced the NSL-KDD dataset to overcome the deficiencies of the KDD Cup dataset. The KDD Cup dataset is the de facto dataset for ML-based IDPS research areas. Thus we use the NSL, as it is an improved version of the KDD Cup dataset, to evaluate the performance of the different models. However, the KDD Cup dataset and the datasets derived from it are now more than 15 years old. To evaluate the performance of the proposed model as well as other ML models on a more recent dataset, we use the Kyoto 2006 + dataset [41].

The most common and well-accepted statistical methods to evaluate the performance of a learning model are cross-validation and bootstrapping. In this paper, we use 10-fold cross-validation to evaluate the detection performance of the selected detection models as it gives us a better understanding of how these models perform on new datasets [42]. The dataset used is divided into 10 random subsets, where 9 subsets are used for training and the remaining 1 subset is used for testing. This process is repeated iteratively until all subsets are tested. The classification errors of every subset are accumulated, following which the mean absolute error is computed.

Different measures are used to evaluate the performance of each model, namely classification model Accuracy (Acc), Detection Rate (DR) (sensitivity), False Alarm Rate (FAR), and Matthew's correlation coefficient (Mcc). These measures are derived from the confusion matrix presented in Table 1. In addition, the Time-to-Build Model (TBM) or training time and Testing Time (TT) are also measured.

Acc reveals a classifier's ability to correctly classify normal and botnet traffic DR is the number of intrusion traffic detected by the model divided by the total number of intrusions in the test set, FAR refers to the percentage of normal traffic classified as intrusion, and Mcc is a correlation coefficient between the observed and detected binary classifications. Mcc has a value between −1 and +1, where a coefficient of +1 represents a perfect detection, 0 implies the detection is no better than random detection, and −1 indicates total disagreement between detection and

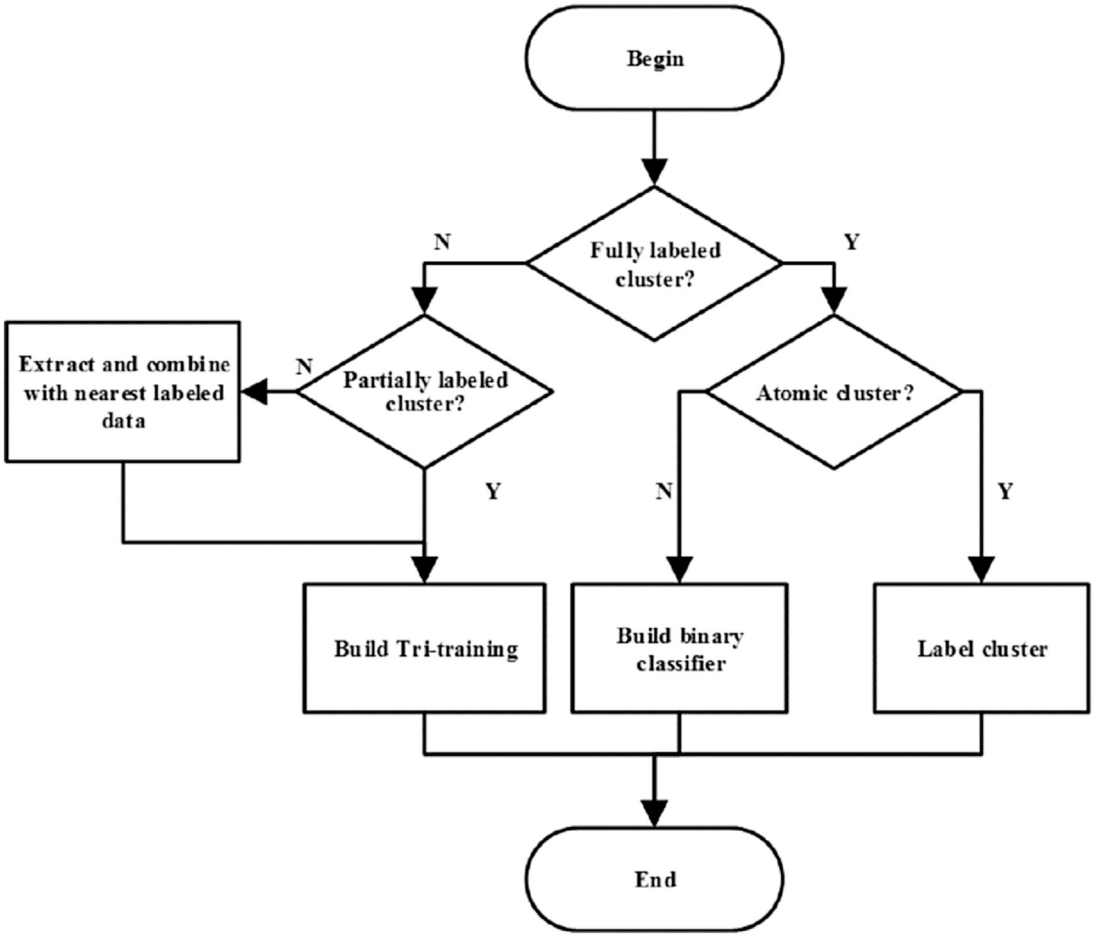

Fig. 3. Flowchart for building local learning models at each layer of SMLC.

Table 1
Confusion matrix.

Actual	Predicted Attack	Predicted Normal
Attack	TP	FN
Normal	FP	TN

observation. A high absolute value of Mcc implies a more robust detection model. The above measures can be defined as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$DR = \frac{TP}{TP + FN} \tag{12}$$

$$FAR = \frac{FP}{TN + FP} \tag{13}$$

$$Mcc = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{14}$$

where True Positive (TP) is the number of correctly classified intrusions, True Negative (TN) is the number of correctly classified normal traffic, False Negative (FN) is the number of intrusions incorrectly classified as normal traffic, and False Positive (FP) is the number of normal traffic incorrectly classified as intrusions. Recent studies focus on training and testing time as indicators of the computational efficiency of the prediction model [43,44,45]. Training and testing time are very important measures of IDS. With ever-increasing network traffic and evolving novel attacks, it is important to frequently update the detection model of the IDS in order to be able to detect novel attacks. As such, a low training time is crucial since it represents the time required to update the detection model. It is desirable that we are able to update the detection model as quickly as possible. Testing time indicates the time required to classify a new incoming instance. Testing is online and produces an events-per-second metric that is a major consideration in the IDS industry [46]. Although the calculation of the absolute training and testing time is platform dependent, the performances of different detection models on the same platform are strong indicators of the relative performances of these models [46]. We aim for high Acc, DR, and Mcc as well as low TBM, FAR, and TT. In addition to the aforementioned measures, we perform a paired statistical significance test to compare the performance of the proposed model with the performance of the tri-training model used in Ref. [47].

We compare the prediction capabilities of the SMLC model with well-known supervised ensemble models, namely RandomForest (RF) [48], Bagging [12], and AdaBoostM1 [13], and the semi-supervised learning model, tri-training [36], used in Ref. [47]. The results of SMLC were obtained by modifying the Java code of Weka package [49].

4.1. Parameter tuning of SMLC

The performance of the proposed model is constrained by two parameters: the number of layers (L) and the number of clusters (K). We have studied the performance of SMLC model while varying L and K and selected the values of L and K that maximize the detection accuracy of the SMLC. Note that, in this study, we use the same value of K in all layers for simplicity. However, it is possible to use different values of K at different layers. Although the number of clusters is empirically selected, the Davies-Bouldin Index (DBI) might be used to select the optimal number of clusters. DBI is a metric to evaluate clustering algorithms [50]. DBI considers the separation between different clusters and the scatter within each cluster. It is defined as the ratio of scatter of cluster i and the separation between cluster i and other clusters. The lower the value of DBI, the better is the separation between clusters and dense clusters.

The final decision of the proposed SMLC is derived from the individual decisions at different layers, aiming to eliminate the uncorrelated errors among base classifiers, which requires that each individual classifier commits its error independently from other base classifiers [51,52]. The Kohvai-Wolpert (KW) variance [53] has been used in the literature to compute the diversity, and it represents the correlation among base classifiers of an ensemble model [54]. The parameter tuning process might include the determination of the value of K that minimizes DBI and then find the value of L that maximizes the KW variance.

4.2. Comparative performance analysis

We compare the performance of the proposed SMLC model with that of the semi-supervised tri-training model used in Srihari and Anitha's work [47] by varying the Percentage of Labeled Data (PLD) in the training dataset from 90% to 10%. For example, a PLD value of 90% means that 90% of the data is labeled and 10% is unlabeled. In each run of the cross-validation method, we partition the training data randomly into $T_{Labeled}$ and $T_{Unlabeled}$ with a PLD value, which is the number of instances in $T_{Labeled}$ as a percentage of the number of all training instances. The PLD value determines how many labeled instances would be used in the training data. Then, the labels of all instances in $T_{Unlabeled}$ are removed to generate unlabeled instances while the labels of all instances in T_{Labled} are kept. We perform the experiment at PLD values of 10%, 30%, 50%, 70%, and 90% for NSL and Kyoto2006 +. In addition, we evaluate the performance of AdaBoostM1, Bagging, and RF when both datasets are fully labeled (*i.e.*, PLD of 100%). The subsequent subsections present the performance of different models on NSL and Kyoto2006 +, respectively.

Table 2
Features of the NSL dataset [24].

	#	Name	Data Type
Basic features	1	duration	Continuous
	2	protocol_type	Symbolic
	3	service	Symbolic
	4	flag	Symbolic
	5	src_bytes	Continuous
	6	dst_bytes	Continuous
	7	land	Symbolic
	8	wrong_fragment	Continuous
	9	urgent	Continuous
	10	hot	Continuous
Content features	11	num_failed_logins	Continuous
	12	logged_in	Symbolic
	13	num_compromised	Continuous
	14	root_shell	Continuous
	15	su_attempted	Continuous
	16	num_root	Continuous
	17	num_file_creations	Continuous
	18	num_shells	Continuous
	19	num_access_files	Continuous
	20	num_outbound_cmds	Continuous
	21	is_hot_login	Symbolic
	22	is_guest_login	Symbolic
Time based traffic features	23	Count	Continuous
	24	srv_count	Continuous
	25	serror_rate	Continuous
	26	srv_error_rate	Continuous
	27	rerror_rate	Continuous
	28	srv_rerror_rate	Continuous
	29	same_srv_rate	Continuous
	30	diff_srv_rate	Continuous
	31	srv_diff_host_rate	Continuous
Host based traffic features	32	dst_host_count	Continuous
	33	dst_host_srv_count	Continuous
	34	dst_host_same_srv_rate	Continuous
	35	dst_host_diff_srv_rate	Continuous
	36	dst_host_same_src_port_rate	Continuous
	37	dst_host_srv_diff_host_rate	Continuous
	38	dst_host_serror_rate	Continuous
	39	dst_host_srv_serror_rate	Continuous
	40	dst_host_rerror_rate	Continuous
	41	dst_host_srv_rerror_rate	Continuous

Table 3
Performance of semi-supervised models on the NSL dataset.

PLD%	Model	Acc%	DR	FAR	Mcc	TBM(*s*)	TT(*s*)
90	SMLC	**99.58927**	**0.99507**	**0.00335**	**0.99177**	**16.88850**	0.96825
	Tri-training	99.51722	0.99432	0.00404	0.99033	83.63010	**0.02111**
	Statistical Significance	***	***	***	***	***	***
70	SMLC	**99.51588**	**0.99467**	**0.00439**	**0.99030**	**26.37443**	0.43307
	Tri-training	99.47884	0.99409	0.00457	0.98956	54.02279	**0.02025**
	Statistical Significance	***	***	*	***	***	***
50	SMLC	**99.45461**	**0.99409**	**0.00503**	**0.98908**	**7.91235**	0.71481
	Tri-training	99.39535	0.99352	0.00565	0.98789	27.94686	**0.01970**
	Statistical Significance	***	**	**	***	***	***
30	SMLC	**99.36371**	**0.99289**	**0.00567**	**0.98726**	**4.49946**	0.67536
	Tri-training	99.19875	0.99214	0.00815	0.98395	12.94706	**0.02040**
	Statistical Significance	***	***	***	***	***	***
10	SMLC	**99.02637**	**0.98910**	**0.00866**	**0.98050**	**1.86190**	0.74932
	Tri-training	98.93817	0.98876	0.01004	0.97873	3.63206	**0.01858**
	Statistical Significance	***	*	***	***	***	***

*** denotes a high significance level (p-value $\leq$ 0.05), ** denotes a low significance level (0.05 < p-value $\leq$ 0.15), and * denotes no significant difference (0.15 < p-value).

4.3. Experiment 1: performance of different models on NSL

The NSL dataset is comprised of 41 features, as in the KDD Cup 99 dataset, and a class label (normal/attack). The features of the NSL dataset are categorized into [40]: (1) basic features including all the features that can be extracted from a Transmission Control Protocol (TCP)/Internet Protocol (IP) connection, (2) traffic features including features that can be computed with respect to a window interval, and (3) content features that inspect the payload of TCP connections. Table 2 presents descriptions of the features of NSL. The dataset contains 148,517 instances, of which 71,463 attacks and 77,054 are normal sessions.

Table 3 summarizes the performance of SMLC against the tri-training model on the NSL dataset. SMLC was given the following parameters for the corresponding PLD values: $L = \{15,9,11,11,12\}$ and $K = \{10,4,10,10,10\}$. The C4.5 (*i.e.*, J48) decision tree was used as a binary classifier in SMLC and the tri-training model. As can be seen in Table 3, the detection accuracy of both models improves as PLD increases. From Table 3, it is evident that SMLC generally outperforms the tri-training model used in Ref. [47] on the NSL dataset. However, the tri-training model requires less testing time to classify a test instance because the classification process of a test instance in the tri-training process involves the decisions of the three constituent classifiers of the tri-training model. On the other hand, SMLC incorporates the decisions of all layers (normally more than 3), which requires finding the nearest cluster to the test instance and the decision of the corresponding classifier at each layer. Majority voting is then employed to determine the final prediction of the test instance. Noticeably, SMLC is able to achieve a detection accuracy comparable to that of the tri-training model by using a lower PLD. For example, SMLC can achieve a detection accuracy of 99.5159% with a PLD of 70%, whereas the tri-training model requires a PLD $\geq$ 90% to achieve a comparable detection accuracy. Note that the TBM of SMLC does not include the time required to perform data clustering. This is because data clustering can be performed online as there is an online version of the K-Means algorithm. However, the TT of SMLC includes the time required to find the nearest cluster at each layer, the time required to find the decision of each corresponding classifier at each layer, and the time required to determine the final decision through majority voting. To show the significance of the proposed approach, we compared the percentage of error reduction using SMLC with that using the tri-training model. For example, when the PLD value is 90%, the error of tri-training ($e_{Tri-training}$) is 100–99.51723 = 0.48277 and the error of SMLC (e_{SMLC}) is 100–99.58927 = 0.41072, error reduction ($error_{reduction}$) is calculated as follows:

$$error_{\text{reduction}} = \frac{e_{\text{Tri-training}} - e_{\text{SMLC}}}{e_{\text{Tri-training}}} \quad (15)$$

The error reduction at a PLD value of 90% is 14.92329%. Fig. 4 shows the error reduction at different PLD values. As shown in Fig. 4, SMLC reduced the detection error by more than 20% compared to the tri-training model by using a PLD value of 30%. Fig. 4 shows that the proposed approach can significantly reduce the detection error compared to the tri-training model at different PLD values.

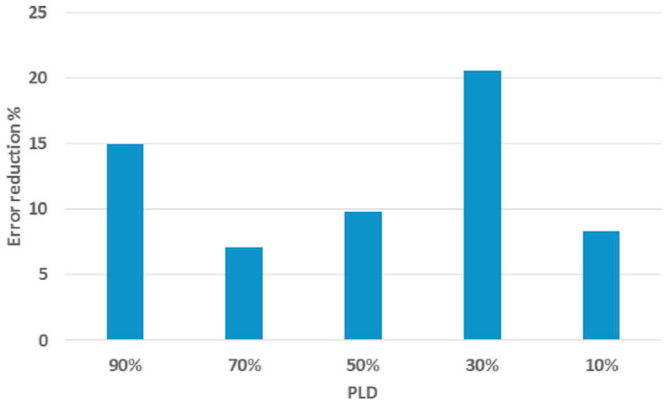

Fig. 4. Error reduction of detection accuracy using the SMLC at different values of PLD of the NSL dataset.

In order to compare the performance of SMLC with that of supervised ensemble models, the performance of the supervised models should be evaluated. Thus we evaluate the performance of the supervised ensemble models on the NSL dataset. Table 4 shows the performance of supervised ensemble models on the NSL dataset. RF has achieved the best detection accuracy among all models. However, it has a relatively high TBM and TT of 59.045s and 0.374s, respectively. The performance of AdaBoostM1 is poor in comparison to that of other models as it achieved a detection accuracy of 94.20874%. However, AdaBoostM1 requires a lower TBM and TT than Bagging and RF. Bagging achieved a moderate performance between the performance of AdaBoostM1 and RF. Tables 3 and 4 show that it is possible to achieve a performance comparable to that of the supervised ensemble models using the SMLC with a partially labeled data. For example, SMLC achieves a detection accuracy of 99.0264% with training data having a PLD value of 10%, which is higher than detection accuracy of AdaBoostM1 with fully labeled training data (*i.e.*, PLD of 100%).

4.4. Experiment 2: performance of different models on the Kyoto2006+

The Kyoto 2006 + dataset contains real network traffic data collected from Nov 2006 to Aug 2009. It has 93,076,270 sessions, of which 50,033,015 are normal sessions, 42,617,536 are known attack sessions, and 425,719 are unknown attack sessions.

Table 4
Performance of different models on the NSL dataset when the training dataset is fully labeled (*i.e.*, PLD of 100%).

Model	Acc%	DR	FAR	Mcc	TBM(*s*)	TT(*s*)
AdaBoostM1	94.20874	0.914879	0.03268	0.88483	**18.54300**	**0.01600**
Bagging	99.55830	0.99514	0.00401	0.99115	27.45000	0.02800
RF	**99.62294**	**0.99545**	**0.00305**	**0.99245**	59.04500	0.37400

Table 5
Features of the Kyoto 2006 + dataset.

#	Name	Description
1.	Duration	The length (seconds) of the connection
2.	Service	The connection's service type, *e.g.*, http, telnet
3.	Source bytes	The number of data bytes sent by the source IP address
4.	Destination bytes	The number of data bytes sent by the destination IP address
5.	Count	The number of connections with the same source IP address and destination IP address as those of the current connection in the past 2 s
6.	Same srv rate	% of connections to the same service in Count feature
7.	Serror rate	% of connections that have SYN errors in Count feature
8.	Srv serror rate	% of connections that have SYN errors in Srv count (the number of connections with the same service type as that of the current connection in the past 2 s) feature
9.	Dst host count	Among the past 100 connections with the same destination IP address as that of the current connection, the number of connections with the same source IP address as that of the current connection
10.	Dst host srv count	Among the past 100 connections with the same destination IP address as that of the current connection, the number of connections with the same service type as that of the current connection
11.	Dst host same src port rate	% of connections with the same source port as that of the current connection in Dst host count feature
12.	Dst host serror rate	%of connections that have SYN errors in Dst host count feature
13.	Dst host srv serror rate	% of connections that SYN errors in Dst host srv count feature
14.	Flag	The state of the connection at the time the connection was written

Known attack sessions refer to network sessions that triggered IDS alarms. On the other hand, unknown attack sessions refer to network sessions that did not trigger an IDS alarm but contained shellcodes. Each session has 24 attributes, of which 14 attributes are derived from the attributes of the KDD Cup 99 dataset and represent the most significant and essential characteristics of a network session. The attributes of contents are excluded because they are not suitable for a Network Intrusion Detection System (NIDS) and domain knowledge is required to extract them. Ten additional attributes, which can be used for further analysis and evaluation of NIDS and for determining the type of attacks in the network, were extracted and added to the attributes of the dataset. As our main objective is to detect attacks regardless of their type (known or unknown attack), we consider the fourteen essential attributes and give the known and unknown attacks the same label. Table 5 presents descriptions of the attributes of the Kyoto 2006 + dataset, excluding the class label. We randomly selected a sample dataset of approximately one million instances while maintaining the original class distribution of the Kyoto 2006 + dataset. The selected sample contains 540,149 normal sessions and 464,518 attacks. This experiment was conducted using the selected sample dataset of the Kyoto2006 + dataset.

Table 6 compares the performance of SMLC with that of the tri-training model on the sample dataset selected from the Kyoto2006 + dataset. SMLC was given the following parameters for the corresponding PLD values: $L = \{10,10,14,15,7\}$ and $K = \{9,8,9,7,7\}$. The C4.5 (*i.e.*, J48) decision tree was used as a binary classifier in SMLC and the tri-training model. As can be seen in Table 6, generally, the detection accuracy of both models improves as PLD increases. From Table 6, the statistical significance test proves that SMLC outperforms the tri-training model used in Ref. [47]. More precisely, SMLC maintains the same performance as, or outperforms, the tri-training model in all performance measures except the testing time, as the tri-training model requires less time to classify a test instance as either normal or attack regardless of the percentage of labeled instances used (*i.e.*, PLD). As mentioned earlier, this is because the SMLC predicts the class label of the test instance by majority voting among all individual decisions of all layers. As has been observed before, SMLC has the ability to achieve higher detection accuracy using a lower PLD. For example, SMLC can achieve a detection accuracy of 99.37711% with a PLD of 50%, whereas the tri-training model requires PLD $\geq$ 70% to achieve the same detection accuracy. This is because SMLC improves the detection accuracy of the base classifiers by building local learning models with the resultant clusters of the K-Means algorithm, leading to a reduction in the errors due to data labeling, and it can capably infer decision boundaries of overlapping data patterns (*i.e.*, non-atomic clusters). In addition, SMLC improves the overall detection accuracy by identifying easy- and difficult-to-classify instances and employing majority voting. In principle, it is possible to maintain a high detection performance while decreasing the cost of data labeling, promoting the deployment of ML-based IDS in real life.

Table 7 presents the performance of supervised ensemble models on the sample dataset selected from the Kyoto2006 + dataset. As can be seen

Table 6
Performance of semi-supervised models on the Kyoto 2006 + dataset.

PLD%	Model	Acc%	DR	FAR	Mcc	TBM(*s*)	TT(*s*)
90	SMLC	**99.39144**	**0.99721**	**0.00892**	**0.98779**	**97.96862**	2.25469
	Tri-training	99.38328	0.99712	0.00899	0.98763	383.78634	**0.07562**
	Statistical Significance	***	***	***	***	***	***
70	SMLC	**99.38099**	**0.99722**	0.00912	**0.98758**	**75.47582**	2.12510
	Tri-training	99.36924	0.99695	**0.00911**	0.98734	243.67701	**0.07493**
	Statistical Significance	***	***	*	***	***	***
50	SMLC	**99.37711**	**0.99725**	**0.00922**	**0.98751**	**49.68831**	3.12047
	Tri-training	99.36198	0.99693	0.00923	0.98719	150.65489	**0.07309**
	Statistical Significance	***	***	*	***	***	***
30	SMLC	**99.35561**	**0.99704**	0.00944	**0.98707**	**26.03816**	2.99768
	Tri-training	99.33958	0.99669	0.00944	0.98675	71.64776	**0.07392**
	Statistical Significance	***	***	*	***	***	***
10	SMLC	**99.29250**	**0.99691**	**0.01049**	**0.98581**	11.83715	1.34672
	Tri-training	99.27528	0.99681	0.01074	0.98547	**10.99234**	**0.06888**
	Statistical Significance	***	*	***	***	*	***

*** denotes a high significance level (p-value $\leq$ 0.05), ** denotes a low significance level (0.05 < p-value $\leq$ 0.15), and * denotes no significant difference (0.15 < p-value).

Table 7
Performance of different models on the Kyoto 2006 + dataset when the training dataset is fully labeled (*i.e.*, PLD of 100%).

Model	Acc%	DR	FAR	Mcc	TBM(*s*)	TT(*s*)
AdaBoostM1	95.88769	0.97128	0.05179	0.91786	**67.44700**	**0.09300**
Bagging	**99.39263**	**0.99710**	0.00880	**0.98781**	205.46500	0.14200
RF	99.37681	0.99667	**0.00872**	0.98749	535.65500	2.54100

in Table 7, Bagging achieves the highest detection accuracy among all the models. It achieves a detection accuracy of 99.39263%, outperforming AdaBoostM1 and RF. Rf and AdaBoostM1 achieve detection accuracies of 99.37681% and 95.88769%, respectively. Tables 6 and 7 indicate that it is possible to achieve a detection accuracy comparable to that of supervised ensemble models using SMLC with 70% labeled training data.

With the rapid growth of data volumes, not only the detection accuracy but also the efficiency and scalability are important. Although, in this paper, the implementation of the proposed model was performed on a single machine, conceptually, SMLC has the potential to efficiently handle large volumes of data by distributing its computational cost among multiple devices as it provides scalable infrastructure for processing a large amount of data on a distributed computing system consisting of a large number of processing nodes.

5. Conclusion

In this paper, we proposed the SMLC model and evaluated its performance on well-known benchmark datasets, NSL and Kyoto 2006+. SMLC generates multiple randomized layers of K-Means algorithm to improve the diversity among its base classifiers, resulting in more accurate detection. The results of the experiments show that SMLC outperforms the semi-supervised tri-training model while using a lower percentage of labeled data denoted by PLD, and achieves a performance comparable to that of well-known ensemble models. The high detection capability and the low cost reflected by the low PLD make SMLC preferable for real-world IDPS tasks. This can be seen as a significant contribution as it bridges the gap between the studies on ML-based IDPS and its practical deployment. In addition, SMLC has the potential to efficiently handle large volumes of data by distributing its computational cost among multiple devices as it provides scalable infrastructure for processing a large amount of data on a distributed computing system consisting of a large number of processing nodes. However, SMLC has a relatively high testing time. Our future works will be to study the scalability of SMLC, automation of its parameter tuning process and reduction of its testing time.

Acknowledgements

This publication was made possible with the support of the ICT Fund (Grant #: G00000103). The statements made herein are solely the responsibility of the authors.

References

[1] O.Y. Al-Jarrah, P.D. Yoo, S. Muhaidat, G.K. Karagiannidis, K. Taha, Efficient machine learning for big data: a review, Big Data Res. 2 (3) (2015) 87–93 big Data, Analytics, and High-Performance Computing, https://doi.org/10.1016/j.bdr.2015.04.001.
[2] S. Abt, H. Baier, A plea for utilising synthetic data when performing machine learning based cyber-security experiments, in: Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, ACM, 2014, pp. 37–45.
[3] O.Y. Al-Jarrah, O. Alhussein, P.D. Yoo, S. Muhaidat, K. Taha, K. Kim, Data randomization and cluster-based partitioning for botnet intrusion detection, IEEE Trans. Cybern. 46 (8) (2016) 1796–1806.
[4] R. Sommer, V. Paxson, Outside the closed world: on using machine learning for network intrusion detection, in: IEEE Symposium on Security and Privacy (SP), IEEE, 2010, pp. 305–316.
[5] T.S. Barhoom, R.A. Matar, Network intrusion detection using semisupervised learning based on normal behaviour's standard deviation, Int. J. Adv. Res. Comput. Commun. Eng. 4 (1) (2015) 375–382.
[6] C. Zhang, Y. Ma, Ensemble Machine Learning, Springer, 2012.
[7] T. Mitchell, Machine Learning, McGraw-Hill International Editions, McGraw-Hill, 1997. https://books.google.ae/books?id=EoYBngEACAAJ.
[8] P. Zhang, C. Zhou, P. Wang, B.J. Gao, X. Zhu, L. Guo, E-tree: an efficient indexing structure for ensemble models on data streams, IEEE Trans. Knowl. Data Eng. 27 (2) (2015) 461–474.
[9] D. Opitz, R. Maclin, Popular ensemble methods: an empirical study, J. Artif. Intell. Res. 11 (1999) 169–198.
[10] Z. Yu, L. Li, J. Liu, G. Han, Hybrid adaptive classifier ensemble, IEEE Trans. Cybern. 45 (2) (2015) 177–190.
[11] B. Tang, M.I. Heywood, M. Shepherd, Input partitioning to mixture of experts, in: Proceedings of the 2002 International Joint Conference on Neural Networks, 2002. IJCNN'02, vol. 1, IEEE, 2002, pp. 227–232.
[12] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[13] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Computational Learning Theory, Springer, 1995, pp. 23–37.
[14] O. Chapelle, B. Schlkopf, A. Zien, Semi-supervised Learning, first ed., The MIT Press, 2010.
[15] F. Breve, L. Zhao, Semi-supervised learning with concept drift using particle dynamics applied to network intrusion detection data, in: BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), IEEE, 2013, pp. 335–340.
[16] S.K. Wagh, S.R. Kolhe, Effective intrusion detection system using semi-supervised learning, in: International Conference on Data Mining and Intelligent Computing (ICDMIC), IEEE, 2014, pp. 1–5.
[17] C. Chen, Y. Gong, Y. Tian, Semi-supervised learning methods for network intrusion detection, in: IEEE International Conference on Systems, Man and Cybernetics, 2008. SMC 2008, IEEE, 2008, pp. 2603–2608.
[18] Y. Li, Z. Li, R. Wang, Intrusion detection algorithm based on semi-supervised learning, in: 2011 International Conference on Information Technology, Computer Engineering and Management Sciences (ICM), vol. 2, IEEE, 2011, pp. 153–156.
[19] C.-Y. Chiu, Y.-J. Lee, C.-C. Chang, W.-Y. Luo, H.-C. Huang, Semi-supervised learning for false alarm reduction, in: Advances in Data Mining. Applications and Theoretical Aspects, Springer, 2010, pp. 595–605.
[20] Y. Meng, et al., Intrusion detection using disagreement-based semi-supervised learning: detection enhancement and false alarm reduction, in: Cyberspace Safety and Security, Springer, 2012, pp. 483–497.
[21] X. Zhu, Semi-supervised Learning Literature Survey.
[22] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, ACM, 1998, pp. 92–100.
[23] S. Fitriani, S. Mandala, M.A. Murti, Review of semi-supervised method for intrusion detection system, in: 2016 Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast), 2016, pp. 36–41, https://doi.org/10.1109/APMediaCast.2016.7878168.
[24] R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas, Y.-L. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Inf. Sci. 378 (2017) 484–497.
[25] Q.-S. Zhu, Q. Fang, A semi-supervised intrusion detection algorithm based on natural neighbor, in: Information System and Artificial Intelligence (ISAI), 2016 International Conference on, IEEE, 2016, pp. 423–426.
[26] Z. Xiang, Z. Xiao, D. Wang, H.M. Georges, Incremental semi-supervised kernel construction with self-organizing incremental neural network and application in intrusion detection, J. Intell. Fuzzy Syst. 31 (2) (2016) 815–823.
[27] N.H. Duong, H.D. Hai, A semi-supervised model for network traffic anomaly detection, in: Advanced Communication Technology (ICACT), 2015 17th International Conference on, IEEE, 2015, pp. 70–75.
[28] A. Mousavi, S.S. Ghidary, Z. Karimi, Semi-supervised intrusion detection via online laplacian twin support vector machine, in: Signal Processing and Intelligent Systems Conference (SPIS), 2015, IEEE, 2015, pp. 138–142.
[29] C.-H. Mao, H.-M. Lee, D. Parikh, T. Chen, S.-Y. Huang, Semi-supervised co-training and active learning based approach for multi-view intrusion detection, in: Proceedings of the 2009 ACM Symposium on Applied Computing, ACM, 2009, pp. 2042–2048.
[30] A. Rahman, B. Verma, Novel layered clustering-based approach for generating ensemble of classifiers, IEEE Trans. Neural Netw. 22 (5) (2011) 781–792.
[31] I. Melnykov, V. Melnykov, On k-means algorithm with the use of mahalanobis distances, Stat. Probab. Lett. 84 (2014) 88–95.
[32] A.A. Ghorbani, I.-V. Onut, Y-means: an autonomous clustering algorithm, in: Hybrid Artificial Intelligence Systems, Springer, 2010, pp. 1–13.
[33] P.-N. Tan, M. Steinbach, V. Kumar, et al., Introduction to Data Mining, vol. 1, Pearson Addison Wesley Boston, 2006.
[34] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[35] J.R. Quinlan, C4. 5: Programs for Machine Learning, Elsevier, 2014.
[36] Z.-H. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.

[37] R. Lippmann, R.K. Cunningham, D.J. Fried, I. Graf, K.R. Kendall, S.E. Webster, M.A. Zissman, Results of the darpa 1998 offline intrusion detection evaluation, in: Recent Advances in Intrusion Detection, vol. 99, 1999, pp. 829–835.
[38] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba, K. Das, The 1999 darpa off-line intrusion detection evaluation, Comput. Netw. 34 (4) (2000) 579–595.
[39] Kdd cup dataset. URL http://kdd.ics.uci.edu/databases/kddcup99.
[40] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on, IEEE, 2009, pp. 1–6.
[41] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, K. Nakao, Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation, in: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, ACM, 2011, pp. 29–36.
[42] R. Kohavi, et al., A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection, in: Ijcai, vol. 14, 1995, pp. 1137–1145.
[43] W.-C. Lin, S.-W. Ke, C.-F. Tsai, Cann: an intrusion detection system based on combining cluster centers and nearest neighbors, Knowledge-Based Syst. 78 (2015) 13–21.
[44] G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, Expert Syst. Appl. 41 (4) (2014) 1690–1700.
[45] G. Nadiammai, M. Hemalatha, Effective approach toward intrusion detection system using data mining techniques, Egypt. Inf. J. 15 (1) (2014) 37–50.
[46] C.A. Peters, Intrusion and Fraud Detection Using Multiple Machine Learning Algorithms, Ph.D. thesis, The University of Manitoba, 2013.
[47] V. Srihari, R. Anitha, Ddos detection system using wavelet features and semi-supervised learning, in: International Symposium on Security in Computing and Communication, Springer, 2014, pp. 291–303.
[48] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[49] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[50] S. Petrovic, A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters, in: Proceedings of the 11th Nordic Workshop of Secure IT Systems, 2006, pp. 53–64.
[51] T. Löfström, Utilizing Diversity and Performance Measures for Ensemble Creation.
[52] K. Tumer, J. Ghosh, Error correlation and error reduction in ensemble classifiers, Connect. Sci. 8 (3–4) (1996) 385–404.
[53] R. Kohavi, D.H. Wolpert, et al., in: Bias Plus Variance Decomposition for Zero-one Loss Functions, vol. 96, ICML, 1996, pp. 275–283.
[54] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[55] D.K. Bhattacharyya, J.K. Kalita, Network Anomaly Detection: a Machine Learning Perspective, CRC Press, 2013.